(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 343 087 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
10.09.2003 Bulletin 2003/37

(51) Int Cl.[7]: **G06F 11/10**

(21) Application number: **03250049.8**

(22) Date of filing: **03.01.2003**

(84) Designated Contracting States:
AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IT LI LU MC NL PT SE SI SK TR
Designated Extension States:
AL LT LV MK RO

(30) Priority: **08.03.2002 US 94086**

(71) Applicant: **Network Appliance, Inc.**
Sunnyvale, California 94089 (US)

(72) Inventors:
- **Kleiman, Steven R.**
  Los Altos, California 94022 (US)
- **English, Robert M.**
  Menlo Park, California 94025 (US)
- **Corbett, Peter F.**
  Lexington, Massachusetts 02420 (US)

(74) Representative: **Collins, John David**
Marks & Clerk,
57-60 Lincoln's Inn Fields
London WC2A 3LS (GB)

(54) Technique for correcting multiple storage device failures in a storage array

(57) A technique efficiently corrects multiple storage device failures in a storage array. The storage array comprises a plurality of concatenated sub-arrays, wherein each sub-array includes a set of data storage devices and a local parity storage device that stores values used to correct a failure of a single device within a row of blocks, e.g., a row parity set, in the sub-array. Each sub-array is assigned diagonal parity sets identically, as if it were the only one present using a double failure protection encoding method. The array further includes a single, global parity storage device holding diagonal parity computed by logically adding together equivalent diagonal parity sets in each of the sub-arrays.

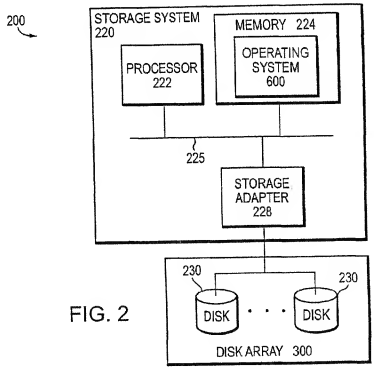

FIG. 2

EP 1 343 087 A2

Description

Field of the Invention

[0001] The present invention relates to arrays of storage systems and, more specifically, to a technique for efficiently reconstructing any one or combination of two failing storage devices of a storage array.

Background of the Invention

[0002] A storage system typically comprises one or more storage devices into which data may be entered, and from which data may be obtained, as desired. The storage system may be implemented in accordance with a variety of storage architectures including, but not limited to, a network-attached storage environment, a storage area network and a disk assembly directly attached to a client or host computer. The storage devices are typically disk drives, wherein the term "disk" commonly describes a self-contained rotating magnetic media storage device. The term "disk" in this context is synonymous with hard disk drive (HDD) or direct access storage device (DASD).

[0003] The disks within a storage system are typically organized as one or more groups, wherein each group is operated as a Redundant Array of Independent (or *Inexpensive)* Disks (RAID). Most RAID implementations enhance the reliability/integrity of data storage through the writing of data "stripes" across a given number of physical disks in the RAID group, and the appropriate storing of redundant information with respect to the striped data. The redundant information enables recovery of data lost when a storage device fails.

[0004] In the operation of a disk array, it is anticipated that a disk can fail. A goal of a high performance storage system is to make the mean time to data loss (MTTDL) as long as possible, preferably much longer than the expected service life of the system. Data can be lost when one or more disks fail, making it impossible to recover data from the device. Typical schemes to avoid loss of data include mirroring, backup and parity protection. Mirroring is an expensive solution in terms of consumption of storage resources, such as disks. Backup does not protect data modified since the backup was created. Parity schemes are common because they provide a redundant encoding of the data that allows for a single erasure (loss of one disk) with the addition of just one disk drive to the system.

[0005] Parity protection is used in computer systems to protect against loss of data on a storage device, such as a disk. A parity value may be computed by summing (usually modulo 2) data of a particular word size (usually one bit) across a number of similar disks holding different data and then storing the results on an additional similar disk. That is, parity may be computed on vectors 1-bit wide, composed of bits in corresponding positions on each of the disks. When computed on vectors 1-bit wide, the parity can be either the computed sum or its complement; these are referred to as even and odd parity respectively. Addition and subtraction on 1-bit vectors are both equivalent to exclusive-OR (XOR) logical operations. The data is then protected against the loss of any one of the disks, or of any portion of the data on any one of the disks. If the disk storing the parity is lost, the parity can be regenerated from the data. If one of the data disks is lost, the data can be regenerated by adding the contents of the surviving data disks together and then subtracting the result from the stored parity.

[0006] Typically, the disks are divided into parity groups, each of which comprises one or more data disks and a parity disk. A parity set is a set of blocks, including several data blocks and one parity block, where the parity block is the XOR of all the data blocks. A parity group is a set of disks from which one or more parity sets are selected. The disk space is divided into stripes, with each stripe containing one block from each disk. The blocks of a stripe are usually at the same locations on each disk in the parity group. Within a stripe, all but one block are blocks containing data ("data blocks") and one block is a block containing parity ("parity block") computed by the XOR of all the data.

[0007] If the parity blocks are all stored on one disk, thereby providing a single disk that contains all (and only) parity information, a RAID-4 implementation is provided. If the parity blocks are contained within different disks in each stripe, usually in a rotating pattern, then the implementation is RAID-5. The term "RAID" and its various implementations are well-known and disclosed in *A Case for Redundant Arrays of Inexpensive Disks (RAID),* by D. A. Patterson, G. A. Gibson and R. H. Katz, Proceedings of the International Conference on Management of Data (SIGMOD), June 1988.

[0008] As used herein, the term "encoding" means the computation of a redundancy value over a predetermined subset of data blocks, whereas the term "decoding" means the reconstruction of a data or parity block by the same process as the redundancy computation using a subset of data blocks and redundancy values. If one disk fails in the parity group, the contents of that disk can be decoded (reconstructed) on a spare disk or disks by adding all the contents of the remaining data blocks and subtracting the result from the parity block. Since two's complement addition and subtraction over 1-bit fields are both equivalent to XOR operations, this reconstruction consists of the XOR of all the surviving data and parity blocks. Similarly, if the parity disk is lost, it can be recomputed in the same way from the surviving data.

[0009] It is common to store the direct XOR sum of data bits as the parity bit value. This is often referred to as "even parity". An alternative is to store the complement of the XOR sum of the data bits as the parity bit value; this is called "odd parity". The use of even or odd parity with respect to the invention disclosed herein is not specified. However, the algorithms referenced here-

in are described as if even parity is used, where such a distinction is relevant. Yet it will be apparent to those skilled in the art that odd parity may also be used in accordance with the teachings of the invention.

[0010] Parity schemes generally provide protection against a single disk failure within a parity group. These schemes can also protect against multiple disk failures as long as each failure occurs within a different parity group. However, if two disks fail concurrently within a parity group, then an unrecoverable loss of data is suffered. Failure of two disks concurrently within a parity group is a fairly common occurrence, particularly because disks "wear out" and because of environmental factors with respect to the operation of the disks. In this context, the failure of two disks concurrently within a parity group is referred to as a "double failure".

[0011] A double failure typically arises as a result of a failure of one disk and a subsequent failure of another disk while attempting to recover from the first failure. The recovery or reconstruction time is dependent upon the level of activity of the storage system. That is, during reconstruction of a failed disk, it is possible that the storage system remains "online" and continues to serve requests (from clients or users) to access (i.e., read and/or write) data. If the storage system is busy serving requests, the elapsed time for reconstruction increases. The reconstruction process time also increases as the size and number of disks in the storage system increases, as all of the surviving disks must be read to reconstruct the lost data. Moreover, the double disk failure rate is proportional to the square of the number of disks in a parity group. However, having small parity groups is expensive, as each parity group requires an entire disk devoted to redundant data.

[0012] Another failure mode of disks is media read errors, wherein a single block or section of a disk cannot be read. The unreadable data can be reconstructed if parity is maintained in the storage array. However, if one disk has already failed, then a media read error on another disk in the array will result in lost data. This is a second form of double failure. A third form of double failure, two media read errors in the same stripe, is unlikely but possible.

[0013] Accordingly, it is desirable to provide a technique that withstands double failures. This would allow construction of larger disk systems with larger parity groups, while ensuring that even if reconstruction after a single disk failure takes a long time (e.g., a number of hours), the system can survive a second failure. Such a technique would further allow relaxation of certain design constraints on the storage system. For example, the storage system could use lower cost disks and still maintain a high MTTDL. Lower cost disks typically have a shorter lifetime, and possibly a higher failure rate during their lifetime, than higher cost disks. Therefore, use of such disks is more acceptable if the system can withstand double disk failures within a parity group.

[0014] A known double failure correcting parity scheme is an EVENODD XOR-based technique that allows a serial reconstruction of lost (failed) disks. EVENODD parity requires exactly two disks worth of redundant data, which is optimal. According to this parity technique, all disk blocks belong to two parity sets, one a typical RAID-4 style XOR computed across all the data disks and the other computed along a set of diagonally adjacent disk blocks. Broadly stated, the disks are divided into blocks of the same size and grouped to form stripes across the disks. Within each stripe, the disk designated to hold parity formed by the set of diagonally adjacent disk blocks is called a *diagonal parity disk* and the parity it holds is called *diagonal parity.* Within each stripe, one block is selected from each of the disks that are not the diagonal parity disk in that stripe. This set of blocks is called a *row parity set* or *"row ".* One block in the row of blocks is selected to hold *row parity* for the row, and the remaining blocks hold data. Within each stripe, one block is selected from each of all but one of the disks that are not the diagonal parity disk in that stripe, with the further restriction that no two of the selected blocks belong to the same row. This is called a *diagonal parity set* or *"diagonal".*

[0015] The diagonal parity sets in the EVENODD technique contain blocks from all but one of the data disks. For *n* data disks, there are *n*-1 rows of blocks in a stripe. Each block is on one diagonal and there are *n* diagonals, each *n*-1 blocks in length. Notably, the EVENODD scheme only works if *n* is a prime number. The EVENODD technique is disclosed in an article of IEEE Transactions on Computers, Vol. 44, No. 2, titled *EVENODD: An Efficient Scheme for Tolerating Double Disk Failures in RAID Architectures,* by Blaum et al, Feb., 1995. A variant of EVENODD is disclosed in U.S. Patent Number 5,579,475, titled *Method and Means for Encoding and Rebuilding the Data Contents of up to Two Unavailable DASDs in a DASD Array using Simple Non-Recursive Diagonal and Row Parity,* by Blaum et al., issued on November 26, 1996.

[0016] The EVENODD technique utilizes a total of *p*+2 disks, *where p* is a prime number and p disks contain data, with the remaining two disks containing parity information. One of the parity disks contains row parity blocks. Row parity is calculated as the XOR of all the data blocks that are at the same position in each of the data disks. The other parity disk contains diagonal parity blocks. Diagonal parity is constructed from *p*-1 data blocks that are arranged in a diagonal pattern on the data disks. The blocks are grouped into stripes of *p*- 1 rows. This does not affect the assignment of data blocks to row parity sets. However, diagonals are constructed in a pattern such that all of their blocks are in the same stripe of blocks. This means that most diagonals "wrap around" within the stripe, as they go from disk to disk.

[0017] Specifically, in an array of *n* x (*n*-1) data blocks, there are exactly *n* diagonals each of length *n*-1, if the diagonals "wrap around" at the edges of the array. The key to reconstruction of the EVENODD parity arrange-

ment is that each diagonal parity set contains no information from one of the data disks. However, there is one more diagonal than there are blocks to store the parity blocks for the diagonals. That is, the EVENODD parity arrangement results in a diagonal parity set that does not have an independent parity block. To accommodate this extra "missing" parity block, the EVENODD arrangement XOR's the parity result of one distinguished diagonal into the parity blocks for each of the other diagonals.

[0018] Fig. 1 is a schematic block diagram of a prior art disk array 100 that is configured in accordance with the conventional EVENODD parity arrangement. Each data block *Dab* belongs to parity sets a and b, where the parity block for each parity set is denoted *Pa*. Note that for one distinguished diagonal (X), there is no corresponding parity set. This is where the EVENODD property arises. In order to allow reconstruction from two failures, each data disk must not contribute to at least one diagonal parity set. By employing a rectangular array of *n* x (*n*-1) data blocks, the diagonal parity sets have *n-1* data block members. Yet, as noted, such an arrangement does not have a location for storing the parity block for all the diagonals. Therefore, the parity of the extra (missing) diagonal parity block (X) is recorded by XOR'ing that diagonal parity into the parity of each of the other diagonal parity blocks. Specifically, the parity of the missing diagonal parity set is XOR'd into each of the diagonal parity blocks P4 through P7 such that those blocks are denoted P4X-P7X.

[0019] For reconstruction from the failure of two data disks, the parity of the diagonal that does not have a parity block is initially recomputed by XOR'ing all of the parity blocks. For example, the sum of all the row parities is the sum of all the data blocks. The sum of all the diagonal parities is the sum of all the data blocks minus the sum of the missing diagonal parity block. Therefore, the XOR of all parity blocks is equivalent to the sum of all the blocks (the row parity sum) minus the sum of all the blocks except the missing diagonal, which is just a parity of the missing diagonal. Actually, *n*-1 copies of the missing diagonal parity are added into the result, one for each diagonal parity block. Since *n* is a prime number, *n*-1 is even, resulting in the XOR of a block with itself an even number of times, which results in a zero block. Accordingly, the sum of the diagonal parity blocks with the additional missing parity added to each is equal to the sum of the diagonal parity blocks without the additional diagonal parity.

[0020] Next, the missing diagonal parity is subtracted from each of the diagonal parity blocks. After two data disks fail, there are at least two diagonal parity sets that are missing only one block. The missing blocks from each of those parity sets can be reconstructed, even if one of the sets is the diagonal for which there is not a parity block. Once those blocks are reconstructed, all but one member of two of the row parity sets are available. This allows reconstruction of the missing member of those rows. This reconstruction occurs on other diagonals, which provides enough information to reconstruct the last missing block on those diagonals. The pattern of reconstructing alternately using row then diagonal parity continues until all missing blocks have been reconstructed.

[0021] Since *n* is prime, a cycle is not formed in the reconstruction until all diagonals have been encountered, hence all the missing data blocks have been reconstructed. If *n* were not prime, this would not be the case. If both parity disks are lost, a simple reconstruction of parity from data can be performed. If a data disk and the diagonal parity disk are lost, a simple RAID-4 style reconstruction of the data disk is performed using row parity followed by reconstruction of the diagonal parity disk. If a data disk and the row parity disk are lost, then one diagonal parity may be computed. Since all diagonals have the same parity, the missing block on each diagonal can be subsequently computed.

[0022] Since each data block is a member of a diagonal parity set, when two data disks are lost (a double failure), there are two parity sets that have lost only one member. Each disk has a diagonal parity set that is not represented on that disk. Accordingly, for a double failure, there are two parity sets that can be reconstructed. EVENODD also allows reconstruction from failures of both parity disks or from any combination of one data disk and one parity disk failure. The technique also allows reconstruction from any single disk failure.

[0023] EVENODD is optimal in terms of the number of disks required; however, disk efficiency for this encoding technique is achieved at the cost of reconstruction performance. EVENODD treats the entire disk array as a single unit. When any disk in the array fails, the system must access all disks in the array to reconstruct the missing blocks. If a single disk fails in an array of n data disks, 1/*n* of the accesses can only be satisfied by reading all *n*-1 remaining disks plus the row parity disk. Accesses to other disks can be satisfied by a single read operation; thus, the average number of accesses per read is 2-1 /*n*. For large *n*, this means that performance of the disk array degrades by a factor of two during reconstruction. In addition, the amount of work the system must do to recover from a failure (and thus the recovery time if the system is constrained) is also proportional to the disk array size. A system with 2n disks takes twice as long to recover as a system with *n* disks. Together, these factors limit the practical size of a RAID group even with protection with multiple disk failures.

Summary of the Invention

[0024] One aspect of the present invention comprises a technique for efficiently correcting multiple storage device failures in a storage array. The storage array comprises a plurality of concatenated sub-arrays, wherein each sub-array includes a set of data storage devices and a local parity storage device that stores parity val-

ues encoded with a single device error correction method used to correct a failure of a single device within a row of blocks, e.g., a row parity set, in the sub-array. Each sub-array is assigned diagonal parity sets identically, as if it were the only one present using a double failure protection encoding method. The array further includes a single, global parity storage device holding diagonal parity computed by logically adding together equivalent diagonal parity sets in each of the sub-arrays.

[0025] According to an aspect of the invention, diagonal parity blocks are computed along the diagonal parity sets of each sub-array. The computed diagonal parity blocks of corresponding diagonal parity sets of the sub-arrays are then logically combined, e.g., using exclusive OR operations, for storage as the diagonal parity on the global parity storage device. The contents of the computed diagonal parity blocks of any sub-array can thereafter be reconstructed by subtracting the combined diagonal parity blocks of the other sub-arrays from diagonal parity stored on the global parity storage device. The global parity storage device can thus be used in connection with the local parity storage devices to correct any double failure within a single sub-array.

[0026] Notably, the double failure protection encoding method used in an embodiment of the invention is independent of the single device error correction method. In addition, there is no restriction on the method used to recover from a single device failure, as long as the method is row-oriented and the rows of blocks in each sub-array are independent, i.e., recovery cannot rely on information from other rows of blocks. The size of these rows need not be related to the size of the rows used to compute diagonal parity if this independence property holds.

[0027] Advantageously, an embodiment of the present invention allows efficient recovery of single failures in an array configured to enable recovery from the concurrent failure of two storage devices within a sub-array of the array. Upon the failure of any data blocks, each in a different sub-array, the embodiment of the invention enables recovery of the data blocks using the single device failure recovery method, e.g., local row parity. Upon the failure of any two blocks within a sub-array, the embodiment of the invention facilitates recovery using a combination of local row parity and global diagonal parity. That is, as long as only one sub-array has a double failure, the data can be recovered because the diagonal parity contributions of the other sub-arrays can be subtracted from the contents of the global parity storage device. In addition, the technique reduces the computation load to compute parity stored in the array during failure-free operation. The technique further reduces the overhead of parity computation, and requires less computation compared to conventional schemes.

BRIEF DESCRIPTION OF THE DRAWINGS

[0028] The above and further advantages of the invention may be better understood by referring to the following description in conjunction with the accompanying drawings in which like reference numerals indicate identical or functionally similar elements:

Fig. 1 is a schematic block diagram of a prior art disk array that is configured in accordance with a conventional EVENODD parity arrangement;

Fig. 2 is a schematic block diagram of an environment including a storage system that may be advantageously used with the present invention;

Fig. 3 is a schematic block diagram of a storage array comprising a plurality of concatenated sub-arrays that may advantageously used with the present invention;

Fig. 4 is a schematic block diagram of a disk array organized in accordance with a row-diagonal (R-D) parity encoding technique;

Fig. 5 is a flowchart illustrating the sequence of steps comprising a novel multiple device failure correcting technique applied to a concatenation of sub-arrays based on R-D encoding in accordance with the present invention; and

Fig. 6 is a schematic block diagram of a storage operating system that may be advantageously used with the present invention.

Detailed Description of an Illustrative Embodiment

[0029] Fig. 2 is a schematic block diagram of an environment 200 including a storage system 220 that may be advantageously used with the present invention. The inventive technique described herein may apply to any type of special-purpose (e.g., file server or filer) or general-purpose computer, including a standalone computer or portion thereof, embodied as or including a storage system 220. Moreover, the teachings of this invention can be adapted to a variety of storage system architectures including, but not limited to, a network-attached storage environment, a storage area network and a disk assembly directly-attached to a client or host computer. The term "storage system" should therefore be taken broadly to include such arrangements in addition to any subsystems configured to perform a storage function and associated with other equipment or systems.

[0030] In the illustrative embodiment, the storage system 220 comprises a processor 222, a memory 224 and a storage adapter 228 interconnected by a system bus 225. The memory 224 comprises storage locations that are addressable by the processor and adapters for storing software program code and data structures associated with an embodiment of the present invention. The

processor and adapters may, in turn, comprise processing elements and/or logic circuitry configured to execute the software code and manipulate the data structures. A storage operating system 600, portions of which are typically resident in memory and executed by the processing elements, functionally organizes the system 220 by, *inter alia*, invoking storage operations executed by the storage system. It will be apparent to those skilled in the art that other processing and memory means, including various computer readable media, may be used for storing and executing program instructions pertaining to the inventive technique described herein.

[0031] The storage adapter 228 cooperates with the storage operating system 600 executing on the system 220 to access information requested by a user (or client). The information may be stored on any type of attached storage array of writeable storage element media such as video tape, optical, DVD, magnetic tape, bubble memory, electronic random access memory, micro-electro mechanical and any other similar media adapted to store information, including data and parity information. However, as illustratively described herein, the information is stored on storage devices such as the disks 230 (HDD and/or DASD) of storage array 300. The storage adapter includes input/output (I/O) interface circuitry that couples to the disks over an I/O interconnect arrangement, such as a conventional high-performance, Fibre Channel serial link topology.

[0032] Storage of information on array 300 is preferably implemented as one or more storage "volumes" that comprise a cluster of physical storage disks 230, defining an overall logical arrangement of disk space. Each volume is generally, although not necessarily, associated with its own file system. The disks within a volume/file system are typically organized as one or more groups, wherein each group is operated as a Redundant Array of Independent (or *Inexpensive)* Disks (RAID). Most RAID implementations enhance the reliability/integrity of data storage through the redundant writing of data "stripes" across a given number of physical disks in the RAID group, and the appropriate storing of parity information with respect to the striped data.

[0033] An embodiment of the present invention comprises a technique for efficiently correcting multiple storage device failures in a storage array having a plurality of concatenated sub-arrays. The inventive technique is preferably implemented by a disk storage layer (shown at 624 of Fig. 6) of the storage operating system 600 to assign diagonal parity sets to each sub-array identically, as if it were the only one present in the array using a double failure protection encoding method. Each sub-array of the storage array includes a set of data storage devices (disks) and a local parity disk that stores parity values encoded with a single device error correction method used to correct a failure of a single disk within a row of blocks, e.g., a row parity set, in the sub-array. The array further includes a single, global parity disk holding diagonal parity.

[0034] Fig. 3 is a schematic block diagram of storage array 300 organized as a plurality of concatenated sub-arrays 310, wherein each sub-array includes a set of data disks (D_1, D_2) and a local parity disk (P_{R1}, P_{R2}). Illustratively, each sub-array 310 is arranged as a concentrated parity, e.g., a RAID-4 style, disk array [A0, A2 ... An] comprising a predetermined number (e.g., seven) of data disks 320 and a row parity disk 330. The cardinality of each sub-array is denoted by Ck (k=0 ... n). To enable recovery from the concurrent failure of two disks in the array, a single diagonal parity disk is provided for the entire array instead of a diagonal parity disk (and row parity disk) for each sub-array. Therefore, the array further includes a global parity disk P_D 350 holding diagonal parity that is computed by the disk storage layer by logically adding together equivalent diagonal parity sets in each of the sub-arrays 310. Double failures within a sub-array can be corrected using only one global diagonal parity disk 350 associated with the entire array. The novel technique thus reduces the number of disks needed to enable efficient recovery from the concurrent failure of two storage devices (disks) in the array.

[0035] According to an embodiment of the invention, diagonal parity blocks are computed along the diagonal parity sets of each sub-array. The computed diagonal parity blocks of corresponding diagonal parity sets of the sub-arrays are then logically combined, e.g., using exclusive OR (XOR) operations, for storage as diagonal parity on the single global parity disk 350. The contents of the computed diagonal parity blocks of any sub-array can thereafter be reconstructed by subtracting the combined diagonal parity blocks of the other sub-arrays from the diagonal parity stored on the global parity disk. The global parity disk can thus be used in connection with the local parity disks to correct any double failure within a single sub-array by noting that, when only one sub-array experiences a double failure, the other sub-arrays are essentially immaterial.

[0036] Notably, the double failure protection encoding method used in an embodiment of the invention is independent of the single device error correction method. In addition, there is no restriction on the method used to recover from a single disk failure (i.e., it need not be "row parity"), as long as the method is row-oriented and the rows of blocks in each sub-array are independent, i.e., recovery cannot rely on information from other rows of blocks. The size of these rows need not be related to the size of the rows used to compute diagonal parity if this independence property holds.

[0037] In the illustrative embodiment, each sub-array 310 is treated as if it were configured with a number of disks equal to a largest sub-array rounded up to a convenient prime number p by assuming any missing disks are zero. Each sub-array further contains *p*-1 rows of blocks. The novel multiple device failure correcting technique can preferably handle a (*m***p*-1)x(*p*-1)) array of blocks, where *m* is any positive integer. Moreover, concatenation of the sub-arrays is based on "row-diagonal"

double failure protection encoding, although other double failure protection encoding methods, such as conventional EVENODD (EO) encoding, may be used with the present invention.

[0038] Row-diagonal (R-D) encoding is a parity technique that provides double failure parity correcting recovery using row and diagonal parity in a disk array. Two disks of the array are devoted entirely to parity while the remaining disks hold data. The contents of the array can be reconstructed entirely, without loss of data, after any one or two concurrent disk failures. An example of a R-D parity technique that may be advantageously used with an embodiment of the present invention is disclosed in the co-pending and commonly-owned European Patent Application titled *Row-Diagonal Parity Technique for Enabling Efficient Recovery from Double Failures in a Storage Array.*

[0039] Fig. 4 is a schematic block diagram of a disk array 400 organized in accordance with the R-D parity encoding technique. Assume *n* equals the number of disks in the array, where n = *p*+*1*, and p is a prime number. The first *n*-2 disks (DO-3) hold data, while disk *n*-1 (RP) holds values encoded with a single device correction algorithm, e.g., row parity, for the data disks D0-D3 and disk *n* (DP) holds diagonal parity. The disks are divided into blocks and the blocks are grouped into stripes, wherein each stripe equals *n*-2 (*i.e., p*-1) rows of blocks. The diagonal parity disk stores parity information computed along diagonal parity sets ("diagonals") of the array. The blocks in the stripe are organized into p diagonals, each of which contains *p*-1 blocks from the data and row parity disks, and all but one of which contains a parity block on the diagonal parity disk. In addition, there are *n*-1 diagonals per stripe.

[0040] The data blocks and the row parity blocks are numbered such that each block belongs to a diagonal parity set and, within each row, each block belongs to a different diagonal parity set. The notation $D_{a,b}$ and $P_{a,b}$ denotes the respective contributions of data (D) and parity (P) blocks to specific row (a) and diagonal (b) parity computations. That is, the notation $D_{a,b}$ means that those data blocks belong to the row or diagonal used for purposes of computing row parity a and diagonal parity b, and $P_{a,b}$ stores the parity for row parity set a and also contributes to diagonal parity set b. For example, $P_{0,8} = D_{0,4} \wedge D_{0,5} \wedge D_{0,6} \wedge D_{0,7}$, wherein "^" represents an XOR operator. The notation also includes the row parity block used for purposes of computing the diagonal parity for a particular diagonal, e.g., $P_4 = D_{0,4} \wedge D_{3,4} \wedge D_{2,4} \wedge P_{1,4}$. Note that each of the diagonal parity blocks stored on the diagonal parity disk represents contributions from all but one of the other disks (including the row parity disk) of the array. For example, the diagonal parity block P_4 has contributions from D0 ($D_{0,4}$), D2 ($D_{3,4}$), D3 ($D_{2,4}$) and RP ($P_{1,4}$), but no contribution from D1. Note also that the diagonal parity for diagonal 8 (P_8) is neither computed nor stored on the diagonal parity disk DP.

[0041] Specifically, the diagonal parity blocks on disk DP include the row parity blocks in their XOR computation. In other words, the diagonal parity stored on the disk DP is computed not only in accordance with the contents of the data disks but also with the contents of the row parity disk. Moreover, the diagonal parity disk contains parity blocks for each of the diagonals of a stripe except one. By encoding the diagonal parity blocks as shown in array 400, the system can recover from any two concurrent disk failures despite the missing diagonal parity (P_8). This results from the fact that the row parity blocks are factored into the computations of the diagonal parity blocks stored on the diagonal parity disk DP.

[0042] The recovery (reconstruction process) aspect of the R-D parity technique is invoked when two data disks (or one data disk and a row parity disk) within a sub-array are concurrently lost due to failure. With any combination of two failed data disks (or one data disk and a row parity disk), row parity cannot be immediately used to reconstruct the lost data; only diagonal parity can be used. Given the structure and organization of the array (i.e., the stripe length and stripe depth are not equal) each diagonal does not include (misses) a block from one of the disks. Therefore, when the two data disks are lost, two diagonals have lost only one member, i.e., for each of the two lost disks, there is one diagonal that does not intersect that disk, therefore no block from that diagonal is lost because of the failure of that disk. A diagonal parity block is stored on the diagonal parity disk for all but one diagonal; therefore, reconstruction of at least one, and usually two, of the missing blocks is initiated using diagonal parity.

[0043] Once a missing block is reconstructed, reconstruction of a row may be completed by reconstructing the other missing block on that row using row parity. When that other block is reconstructed, a determination is made as to whether the block belongs to a diagonal for which there is parity, the other missing block on that diagonal can be reconstructed from the other disk that is on that diagonal using diagonal parity. That is, for all but the missing diagonal, once one block on the diagonal is reconstructed, the other can be reconstructed. The other missing block in that row parity set is then reconstructed. However, if the block belongs to a diagonal for which there is no parity (i.e, the missing diagonal), then a determination is made as to whether all blocks have been reconstructed. If not, the pattern of first reconstructing based on diagonal parity, then on row parity, continues until the last data block used in computation of the missing diagonal parity set is reached. Once all blocks have been reconstructed, the reconstruction process is complete.

[0044] Fig. 5 is a flowchart illustrating the sequence of steps comprising the novel multiple device failure correcting technique as applied to storage array 300 having a concatenation of sub-arrays 310 based on R-D encoding. The sequence starts in Step 500 and proceeds to

Step 502 where all sub-arrays A[0-n], including row parity devices (disks) 330, are concatenated such that the total number of data and row parity disks over all Ck is prime. In Step 504, the diagonal parity disk 350 is added to form array 300. In Step 506, the contents of the diagonal parity disk 350 are encoded by computing the diagonal parity of each sub-array according to the R-D parity technique, combining the equivalent diagonal parity computations for each sub-array using XOR operations and storing them on the diagonal parity disk.

[0045] In Step 508, the array fails. If the failure is a single disk failure (Step 510), a determination is made in Step 512 as to whether the failure is to a disk in a sub-array. If so, the failed data or row parity disk is reconstructed in Step 514 using local row parity associated with that sub-array. The sequence then ends in Step 532. If the single failure is not to a disk of a sub-array, the failed global diagonal parity disk is reconstructed using all disks (data and row parity disks) of all sub-arrays of the entire array. This is because the diagonal parity sets (i.e., diagonals) span the entire array of disks. In particular, the diagonal parity stored on the failed global diagonal parity disk 350 is reconstructed in Step 516 by logically combining, e.g., using XOR operations, equivalent diagonal parity sets in the sub-arrays 310. The sequence then ends in Step 532.

[0046] If the failure is not a single disk failure, a determination is made in Step 518 as to whether the array failure is a double failure within a sub-array. If not, a determination is made in Step 520 as to whether the failure includes the diagonal parity disk. If not, each disk failure is either a data or row parity disk failure that occurs in a different sub-array and, in Step 522, the failed disk in each sub-array is reconstructed using local row parity. The sequence then ends in Step 532.

[0047] If one of the failures includes the global diagonal parity disk, then a determination is made is Step 524 as to whether the other failed disk includes a row parity disk. If so, failures to a row parity disk and the diagonal parity disk are reconstructed by first reconstructing the failed row parity disk from the data disks of the sub-array and then reconstructing the diagonal parity disk from equivalent diagonal parity sets in the sub-arrays (Step 526). The sequence then ends in Step 532. If not, failures to a data disk and the diagonal disk are reconstructed by first reconstructing the data disk from local row parity associated with the sub-array and then reconstructing the diagonal parity disk from equivalent diagonal parity sets in the sub-arrays (Step 528). The sequence then ends in Step 532.

[0048] In Step 530, two disk failures (a double failure) within a sub-array are globally recovered using the R-D reconstruction process. Here, two failures occur within disks protected by the same row parity; therefore, diagonal parity is needed for reconstruction. According to the invention, as long as only one sub-array has a double failure, the data can be recovered because the contribution of the other sub-arrays can be subtracted from the diagonal parity. Specifically, the diagonal parity of the non-double failed sub-arrays are subtracted from the contents of the diagonal parity disk and then the data and/or row parity of the failed sub-array are reconstructed using the R-D technique. Note that since the conditions on the diagonal parity disk are generally the same as described with respect to the R-D parity technique, the diagonal parity disk is used to recover at least one data block within the failed sub-array. Once that block is recovered, row parity within the sub-array is used to recover the corresponding block in the other failed disk. This process continues in accordance with the R-D reconstruction process. The sequence then ends in Step 532.

[0049] Note that a difference between the present technique and the R-D technique is the observation that virtually any number of disks in the array may be row parity disks. The row parity disks essentially define sub-arrays within the array. Reconstruction based on local row parity involves only data disks (i.e., row parity sets) of the sub-array.

[0050] Therefore, the inventive correcting technique allows more efficient (and easier) recovery of single failures in array 300 adapted to enable recovery from concurrent failures of two disks within a sub-array.

[0051] The invention further allows adding of a single diagonal parity disk to an existing array of data and row parity disks to thereby provide protection against double failures in the array. The R-D parity reconstruction algorithm may then be applied.

[0052] It should be further noted that the technique described herein is capable of correcting more than two failures in the array 300, provided that there are no more than two failures in any one sub-array, and that there is no more than one sub-array with two failures, and that if there are two failures in any sub-array, that the diagonal parity disk has not also failed. For example, assume there are three sub-arrays, each comprising one or more data disks and a row parity disk. The present invention enables recovery from a single disk (data or row parity) failure within each sub-array and another disk failure anywhere in the array, for a total of four disk failures within the entire array. In the case of two disk failures within a single sub-array, reconstruction begins by locating a diagonal parity set that has lost only one member. That is, reconstruction begins with a missing block from diagonal parity of a diagonal parity set not represented on one of the failed disks. From there, reconstruction of the other missing block in the row parity set can be effected, with the row-diagonal reconstruction procedure continuing until the last data block used in computation of the missing diagonal parity set is reached.

[0053] Advantageously, an embodiment of the present invention allows efficient recovery of single failures in an array configured to enable recovery from the concurrent failure of two storage devices within a sub-array of the array. Upon the failure of any data blocks,

each in a different sub-array, the invention enables recovery of the data blocks using the single device failure recovery method, e.g., local row parity. Upon the failure of any two blocks within a sub-array, an embodiment of the invention facilitates recovery using a combination of local row parity and global diagonal parity. That is, as long as only one sub-array has a double failure, the data can be recovered because the diagonal parity contributions of the other sub-arrays can be subtracted from the contents of the global parity storage device.

[0054] Fig. 6 is a schematic block diagram of the storage operating system 600 that may be advantageously used with the present invention. In the illustrative embodiment, the storage operating system is preferably the NetApp® Data ONTAP™ operating system available from Network Appliance, Inc., Sunnyvale, California that implements a Write Anywhere File Layout (WAFL™) file system. As used herein, the term "storage operating system" generally refers to the computer-executable code operable to perform a storage function in a storage system, e.g., that implements file system semantics and manages data access. In this sense, the ONTAP software is an example of such a storage operating system implemented as a microkernel and including the WAFL layer to implement the WAFL file system semantics and manage data access. The storage operating system can also be implemented, for example, as an application program operating over a general-purpose operating system, such as UNIX® or Windows NT®, or as a general-purpose operating system with storage functionality or with configurable functionality, which is configured for storage applications as described herein.

[0055] The storage operating system comprises a series of software layers, including a media access layer 610 of network drivers (e.g., an Ethernet driver). The operating system further includes network protocol layers, such as the Internet Protocol (IP) layer 612 and its supporting transport mechanisms, the Transport Control Protocol (TCP) layer 614 and the User Datagram Protocol (UDP) layer 616. A file system protocol layer provides multi-protocol data access and, to that end, includes support for the Common Internet File System (CIFS) protocol 618, the Network File System (NFS) protocol 620 and the Hypertext Transfer Protocol (HTTP) protocol 622. In addition, the operating system 600 includes a disk storage layer 624 that implements a disk storage protocol, such as a RAID protocol, and a disk driver layer 626 that implements a disk access protocol such as, e.g., a Small Computer Systems Interface (SCSI) protocol. Bridging the disk software layers with the network and file system protocol layers is a WAFL layer 680 that preferably implements the WAFL file system.

[0056] It should be noted that the software "path" through the storage operating system layers described above needed to perform data storage access for a user request received at the storage system may alternatively be implemented in hardware. That is, in an alternate embodiment of the invention, the storage access request data path 650 may be implemented as logic circuitry embodied within a field programmable gate array (FPGA) or an application specific integrated circuit (ASIC). This type of hardware implementation may increase the performance of the service provided by system 220 in response to a user request. Moreover, in another alternate embodiment of the invention, the processing elements of adapter 228 may be configured to offload some or all of the storage access operations from processor 222 to thereby increase the performance of the service provided by the storage system.

[0057] It is expressly contemplated that the various processes, architectures and procedures described herein can be implemented in hardware, firmware or software. For example, a common embodiment of the invention may comprise software code running on a general or special purpose computer, including an embedded microprocessor. However, it is entirely possible, and in some cases preferred, to implement the invention in a FPGA, an ASIC or in some other hardware or software embodiment. Those skilled in the art will understand that the inventive algorithm described herein can be implemented using a variety of technical means.

[0058] The illustrative embodiments set forth herein are described with respect to a concentrated parity arrangement, where the local parity blocks of each sub-array are all stored on the same disk. In yet another alternate embodiment of the invention, the inventive technique can be utilized in connection with other sub-array organizations, such as a distributed parity arrangement (e.g., RAID-5), where the location of the local parity blocks shifts from disk to disk in the sub-array in different sets of rows. However, a scaling aspect of the present invention (i.e., the ability to add disks to the array without reorganizing existing data and parity blocks in the future) practically applies to only the concentrated parity technique, since the configuration of diagonal parity sets takes into account the existence of "imaginary" (absent) disks having zero-valued blocks. This type of scaling would be quite difficult using a distributed parity arrangement wherein the rotated parity may fall on such imaginary disks.

[0059] An aspect of the present invention operates on sub-arrays having sizes ranging from 2 to p storage devices. That is, by repeating sub-arrays of 2 to p devices, with p-1 rows, the invention provides double failure protection within any sub-array and, hence, in the entire storage array. The proof is that the contents of a "sub-array" diagonal parity device for any one sub-array can be reconstructed by subtracting the computed diagonal parity of the other sub-arrays from the global diagonal parity device for the entire storage array. (Note that the single global diagonal parity device is the addition of the equivalent sub-array diagonal parity devices of the sub-arrays.) An embodiment of the invention requires that the blocking of stripes and the number of devices within each sub-array (other than the diagonal parity device)

meet the constraints of the applicable double failure protection encoding method, as described herein with the R-D (or EO) encoded arrays.

[0060] While there have been shown and described illustrative embodiments for efficiently correcting multiple storage device failures in a storage array, it is to be understood that various other adaptations and modifications may be made within the scope of the invention. For example, in an alternate embodiment, the present invention can be used in the area of communications as a forward error correction technique that enables, e.g., multicast distribution of data over long latency links (e.g., satellite). In this embodiment, the data may be divided into storage elements, such as packets or units of data adapted for transmission over an electronic communications medium (network), with every pth packet containing the row parity XOR of the previous *p-1* packets. A packet containing diagonal parity is sent after every *n* sets of p packets. It will be understood to those skilled in the art that other organizations and configurations of packets may be employed in accordance with the principles of the invention. Note that the row parity packets have to be at least as large as the largest data packet in each sub-group (set) and that the diagonal parity packet must be at least as large as the largest data packet in any sub-group. Also, the minimum diagonal parity packet size is *p-1* bits, where *p* is the smallest prime number that is at least as large as the number of packets in any sub-group of packets. If one packet is dropped in a set of *p*, it is recoverable from the row parity. If two packets are dropped in one set of *p*, recovery may be achieved using diagonal parity.

[0061] The present invention can be implemented as a computer program and thus the present invention encompasses any suitable carrier medium carrying the computer program for input to and execution by a computer. The carrier medium can comprise a transient carrier medium such as a signal e.g. an electrical, optical, microwave, magnetic, electromagnetic or acoustic signal, or a storage medium e.g. a floppy disk, hard disk, optical disk, magnetic tape, or solid state memory device.

Claims

1. A system adapted to correct multiple storage device failures in a storage array, the system comprising:

   a storage array having a plurality of concatenated sub-arrays, each sub-array including a set of data storage devices and a local parity storage device, each sub-array assigned diagonal parity sets identically as if it were the only one present using a double failure protection encoding method, the array further including a global parity storage device holding diagonal parity computed by logically adding together equivalent diagonal parity sets in each of the sub-arrays, wherein the global parity storage device is adapted to be used in connection with the local parity storage device of each sub-array to correct a double failure within the sub-array.

2. The system of Claim 1 wherein the local parity storage device is configured to store values encoded with a single device error correction method used to correct a failure of a single device within a row parity set in the sub-array.

3. The system of Claim 2 wherein the row parity set is a row of blocks.

4. The system of Claim 2 or Claim 3 wherein the encoding method that protects against a second device failure is independent of the single device error correction method.

5. The system of Claim 4 wherein the double failure protection encoding method is row-diagonal encoding.

6. The system of Claim 4 or Claim 5 wherein the single device error correction method is row parity.

7. The system of any preceding claim wherein each sub-array is organized as a concentrated parity device array.

8. The system of any preceding claim wherein each sub-array is organized as a distributed parity device array.

9. The system of any preceding claim wherein the storage devices are video tape, magnetic tape, optical, DVD, bubble memory, electronic random access memory or magnetic disk devices.

10. A method for encoding data for correction of double failures in a storage array, the method comprising the steps of:

    organizing the storage array as a plurality of concatenated sub-arrays, each sub-array including a set of data storage devices and a local parity storage device, the storage array further including a global parity storage device for holding diagonal parity;
    assigning diagonal parity sets to each sub-array identically as if the sub-array were the only one present using a double failure protection encoding method; and
    computing the diagonal parity by logically adding together equivalent diagonal parity sets in each of the sub-arrays.

11. The method of Claim 10 further comprising correcting storage device failure within the array using the local parity storage device associated with each sub-array and the global parity storage device associated with the storage array.

12. The method of Claim 11 wherein the step of computing comprises the step of computing diagonal parity blocks along the diagonal parity sets of each sub-array.

13. The method of Claim 12 wherein the step of computing further comprises the step of logically combining the computed diagonal parity blocks of corresponding diagonal parity sets of the sub-arrays for storage as the diagonal parity on the global parity storage device.

14. The method of Claim 13 wherein the step of logically combining comprises the step of using exclusive OR operations to compute the diagonal parity.

15. The method of Claim 13 or Claim 14 wherein the step of correcting comprises the step of reconstructing the computed diagonal parity blocks of any sub-array by subtracting the combined diagonal parity blocks of the other sub-arrays from the diagonal parity stored on the global parity storage device.

16. The method of any one of Claims 10 to 15 further comprising the step of storing parity values encoded with a single device error correction method on the local parity storage device of each sub-array.

17. The method of Claim 16 wherein the step of correcting further comprises the step of correcting a failure of a single device within a row of blocks in each sub-array using the single device error correction method.

18. The method of Claim 17 wherein the encoding method that protects against a second device failure is independent of the single device error correction method.

19. The method of Claim 18 wherein the single device error correction method is row-oriented and the rows of blocks in each sub-array are independent.

20. The method of any one of Claims 10 to 19 wherein the step of organizing comprises the step of organizing each sub-array as a concentrated parity device array.

21. The method of any one of Claims 10 to 20 wherein the step of organizing comprises the step of organizing each sub-array as a distributed parity device array.

22. The method of any one of Claims 10 to 21 wherein the storage devices are video tape, magnetic tape, optical, DVD, bubble memory, electronic random access memory or magnetic disk devices.

23. Apparatus for correcting double failures in a storage array, the apparatus comprising:

means for organizing the storage array as a plurality of concatenated sub-arrays, each sub-array including a set of data storage devices and a local parity storage device, the storage array further including a global parity storage device for holding diagonal parity;
means for assigning diagonal parity sets to each sub-array identically as if the sub-array were the only one present using a double failure protection encoding method;
means for computing the diagonal parity using parity encoding operations that logically add together equivalent diagonal parity sets in each of the sub-arrays; and
means for correcting storage device failure within the array using parity decoding operations on the local parity storage device associated with each sub-array and the global parity storage device associated with the storage array.

24. The apparatus of Claim 23 wherein the means for organizing comprises means for organizing each sub-array as a concentrated parity device array.

25. The apparatus of Claim 23 wherein the means for organizing comprises means for organizing each sub-array as a distributed parity device array.

26. The apparatus of any one of Claims 23 to 25 wherein the storage devices are video tape, magnetic tape, optical, DVD, bubble memory, electronic random access memory or magnetic disk devices.

27. The apparatus of any one of Claims 23 to 26 wherein the parity encoding and decoding operations are performed by special purpose hardware, such as a field programmable gate array or an application specific integrated circuit.

28. A system adapted to correct multiple storage element failures in a storage array, the system comprising:

a storage array having a plurality of concatenated sub-arrays, each sub-array including a set of data storage elements and a local parity storage element configured to store values encoded with a single element error correction method used to correct a failure of a single el-

ement within a row parity set in the sub-array, each sub-array assigned diagonal parity sets identically as if it were the only one present using a double failure protection encoding method, the array further including a global parity storage element holding diagonal parity computed by logically adding together equivalent diagonal parity sets in each of the sub-arrays, wherein the global parity storage element is used in connection with the local parity storage element of each sub-array to correct a double failure within the sub-array.

29. The system of Claim 28 wherein the storage elements are packets and wherein logically adding together comprises use of exclusive OR (XOR) operations.

30. The system of Claim 29 wherein *p* is a prime number and wherein every *pth* packet contains a row parity XOR of previous p-1 packets.

31. The system of Claim 30 wherein a packet containing diagonal parity is sent after every *n* sets of *p* packets.

32. The system of Claim 31 wherein, if one packet is dropped in a set of *p*, the packet is recoverable using row parity and if two packets are dropped in one set of *p*, the packets are recovered using row and diagonal parity.

33. A method for correcting double failures within data adapted for transmission over a communication medium, the method comprising the steps of:

dividing the data into packets for transmission over the communications medium;
organizing the packets into *n* sub-groups of *p* packets, wherein *p* is a smallest prime number that is at least as large as a number of packets in any sub-group of packets and wherein each sub-group of packets includes data packets and a local parity packet configured to store values encoded with a single error correction method used to correct a failure of a single packet within a row parity set in the sub-group;
assigning diagonal parity sets to each sub-group identically as if it were the only one present using a double failure protection encoding method; and
providing a global parity packet with the group of packets, the global parity packet holding diagonal parity computed by logically adding together equivalent diagonal parity sets in each of the sub-groups, wherein the global parity packet is used in connection with the local parity packet of each sub-group to correct a double failure within the sub-array.

34. The method of Claim 33 wherein logically adding together comprises use of exclusive OR (XOR) operations and wherein everypth packet contains a row parity XOR of previous *p*-1 packets.

35. The method of Claim 34 wherein the step of providing comprises the step of sending the global parity packet containing diagonal parity after every *n* subgroup of *p* packets.

36. The method of Claim 35 further comprising the steps of: if one packet is dropped in a set of *p*, recovering the packet using row parity; and
if two packets are dropped in one set of *p*, recovering the packets using row and diagonal parity.

37. A carrier medium carrying computer implementable code for controlling a computer to carry out the method of any one of Claims 10 to 22 or 33 to 36.

DATA DISK 0	DATA DISK 1	DATA DISK 2	DATA DISK 3	DATA DISK 4	ROW PARITY DISK	DIAGONAL PARITY DISK
D04	D05	D06	D07	D0X	P0	P4X
D15	D16	D17	D1X	D14	P1	P5X
D26	D27	D2X	D24	D25	P2	P6X
D37	D3X	D34	D35	D36	P3	P7X

FIG. 1
(PRIOR ART)

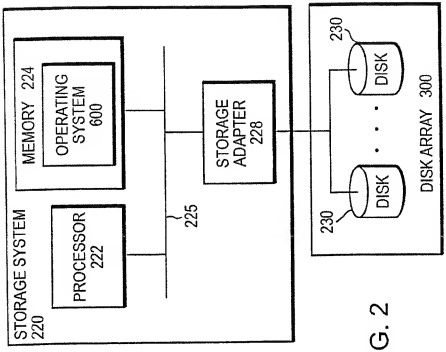
200
STORAGE SYSTEM
220
PROCESSOR
222
MEMORY 224
OPERATING
SYSTEM
600
225
STORAGE
ADAPTER
228
230
DISK
DISK
230
DISK ARRAY 300

FIG. 2

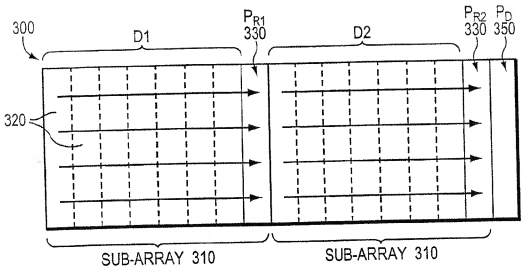
300
D1
P_{R1}
330
D2
P_{R2}
330
P_D
350
320
SUB-ARRAY 310
SUB-ARRAY 310

FIG. 3

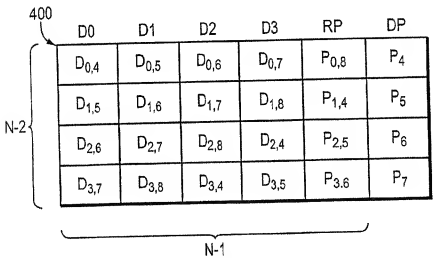
400
D0
D1
D2
D3
RP
DP
$D_{0,4}$ $D_{0,5}$ $D_{0,6}$ $D_{0,7}$ $P_{0,8}$ P_4
$D_{1,5}$ $D_{1,6}$ $D_{1,7}$ $D_{1,8}$ $P_{1,4}$ P_5
$D_{2,6}$ $D_{2,7}$ $D_{2,8}$ $D_{2,4}$ $P_{2,5}$ P_6
$D_{3,7}$ $D_{3,8}$ $D_{3,4}$ $D_{3,5}$ $P_{3.6}$ P_7
N-2
N-1

FIG. 4

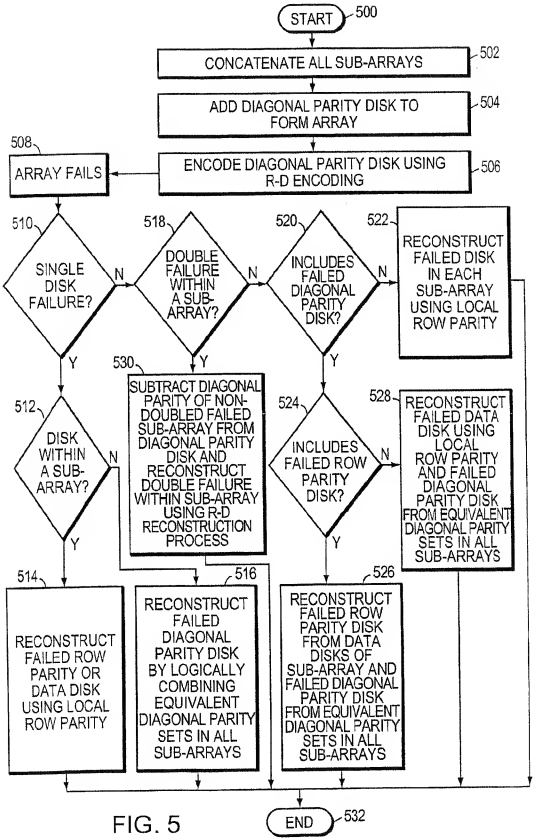
START
500
CONCATENATE ALL SUB-ARRAYS
502
ADD DIAGONAL PARITY DISK TO FORM ARRAY
504
ENCODE DIAGONAL PARITY DISK USING R-D ENCODING
506
ARRAY FAILS
508
SINGLE DISK FAILURE?
510
N
Y
DOUBLE FAILURE WITHIN A SUB-ARRAY?
518
N
Y
INCLUDES FAILED DIAGONAL PARITY DISK?
520
N
Y
RECONSTRUCT FAILED DISK IN EACH SUB-ARRAY USING LOCAL ROW PARITY
522
DISK WITHIN A SUB-ARRAY?
512
N
Y
SUBTRACT DIAGONAL PARITY OF NON-DOUBLED FAILED SUB-ARRAY FROM DIAGONAL PARITY DISK AND RECONSTRUCT DOUBLE FAILURE WITHIN SUB-ARRAY USING R-D RECONSTRUCTION PROCESS
530
INCLUDES FAILED ROW PARITY DISK?
524
N
Y
RECONSTRUCT FAILED DATA DISK USING LOCAL ROW PARITY AND FAILED DIAGONAL PARITY DISK FROM EQUIVALENT DIAGONAL PARITY SETS IN ALL SUB-ARRAYS
528
RECONSTRUCT FAILED ROW PARITY OR DATA DISK USING LOCAL ROW PARITY
514
RECONSTRUCT FAILED DIAGONAL PARITY DISK BY LOGICALLY COMBINING EQUIVALENT DIAGONAL PARITY SETS IN ALL SUB-ARRAYS
516
RECONSTRUCT FAILED ROW PARITY DISK FROM DATA DISKS OF SUB-ARRAY AND FAILED DIAGONAL PARITY DISK FROM EQUIVALENT DIAGONAL PARITY SETS IN ALL SUB-ARRAYS
526
END
532

FIG. 5

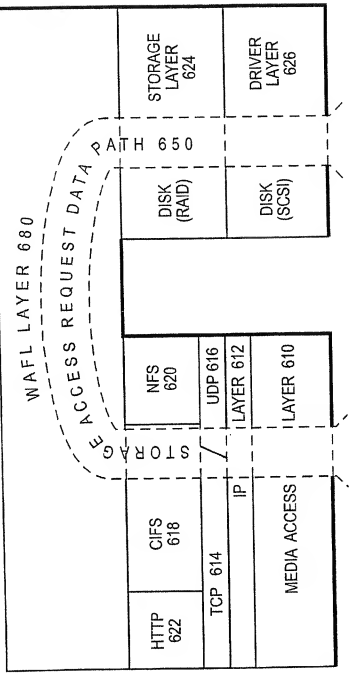
600
WAFL LAYER 680
ACCESS REQUEST DATA PATH 650
STORAGE
HTTP 622
CIFS 618
NFS 620
TCP 614
UDP 616
IP
LAYER 612
MEDIA ACCESS
LAYER 610
DISK (RAID)
DISK (SCSI)
STORAGE LAYER 624
DRIVER LAYER 626

FIG. 6